NIEM-UML Subsetting and Extension Transcript

NIEM Program Management Office Resource

Welcome to NIEM-UML Tutorial on subsetting and extension.We assume you have a little bit of background, that you know the basics of NIEM and UM.Perhaps you've seen the NIEM-UML High-Level Introduction. In this tutorial, we'll cover NIEM information model types, reference models and how to subset them, extensions, and NIEM augmentations.If you need more background, please see the tutorial webinars on the NIEM.gov website.

NIEM defines specific types of models for specific purposes.The most general is a reference model.A reference model contains content that is intended to be reused across multiple exchanges or sets of exchanges. A subset defines just the parts and pieces of a reference model needed for a particular purpose, whereas an extension adds new concepts or specializes existing one. An exchange identifies the particular document types that travel between exchange partners. Information exchanges can be defined using different methodologies, some being more top down, others more bottom-up.In a top methodology, you may start with an extension model representing the high level concepts of your stakeholders. You could then explore the NIEM reference models, find the concepts that correspond to these and subset them for your particular purpose.

NIEM-UML constructs, such as generalization, augmentation, and references are then used to connect the two.In a more bottom up scenario, you may start by exploring the NIEM reference models, find the concepts that are useful, create a subset model for those that you want to use, and then extend those to add the classes and properties you need for your purpose- again using NIEM-UML constructs. Both methodologies will work, as well as combinations of methodologies. The best choice may depend on your particular situation.NIEM information models are defined using UML packages.Packages collect combinations of classes and associations and properties for a particular purpose.These are then stereotyped as a NIEM information model which also includes a Namespace concept.A Namespace defines whether its NIEM conformant, its XML namespace string, and its version.Information models have a tagged value that defines the purpose of the model, whether it's an exchange, an extension, a subset, a reference, etc.These stereotypes and tagged values should be used for any package intended for use with NIEM.For example, if we looked at the PetAdoptionExtension package in UML, we see it's marked as an information model, that its purpose is set to extension, it's defined as being NIEM conformant, it has an XML Namespace and a version.

There are also relationships between information models.One information model may use another, so we see it is common for an exchange to use an extension and an extension to use one or more subsets.A subset model also references its reference model.This means everything in the subset is conformant with concepts already defined in the reference model.NIEM reference models provide the basis for subsetting and reuse.And each of these is available as a NIEM-UML model.There's a whole library of reference models for different domains and different purposes, and the most general being NIEM Core.NIEM Core being reused across all domains and exchanges.

What you do is take a concept out of a reference model.In this example, we're taking PersonType out of NIEM Core.You then define a subset of it that references that type.In this case we see that Person references PersonType.This means that everything defined in Person must match something defined in PersonType.Let's look at an example.We'll find PersonType in NIEM Core, create a subset of it, we'll then load the Emergency Management domain, another reference model, and look for the alert concept.We'll create an Emergency Management subset model and subset this alert concept.Note that your UML tool may look slightly differently and the NIEM PMO does not endorse any particular UML tooling.We already have some of the NIEM reference models loaded.Here we see NIEM Core.You'll notice its quite extensive.We'll search within NIEM Core for something about Person.Since there's so many, I'm going to restrict my search to Classes.You'll note we found 34 Classes having to do with Person.Everything from their hair color, to their skin type, to PersonType in general.What I frequently like to do is take a class that I found and load it into a Scratchpad Diagram to take a look.Here we see the PersonType and the properties of PersonType.There are also some associations from Person, and we might want to take a look at these.

So we're going to ask the tool to look at the related elements, and I'm going to look for the associations from Person within NIEM Core.Now we can see there's quite a bit and I'll ask the tool to lay this out for us.This is one of the mechanisms for exploring the model in UML.We now see that there are not only the properties of Person, but quite a few related Types.Having decided that PersonTYpe is what we're looking for, we're going ot go back and make a subset of it.So what we've done is select PersonType out of NIEM Core into our subset model.Note that it doesn't have any properties or any associations.We can now ask the tool to show us what properties or associations are available.For instance, we may want to know their birthdate, their name, and social security number.Notice that it brings in these properties.If we look at the NIEM model, we also see that it's necessary to bring in other types from NIEM Core- the date type, the identification type, person name.So we'll also want to select the properties that we need in each of these.For instance, maybe we only need their full name.In this way, we build up a subset of NIEM Core designed for our purpose.But we don't want to limit ourselves to NIEM Core.Say that we also want to include the Emergency Management (EM) domain.You see that the NIEM reference models are readily available.We'll select the Emergency Management domain and load that into our model.Now we see that the Emergency Management domain is available as a package.We can now explore the contents of the Emergency Management domain, not as big as NIEM Core.

Let's also go back to our Scratchpad and look at this AlertType to see if this is has the kind of thing that we're looking for.We think it does, so what we'd like to do is create a subset of the EM domain in our project.Now for NIEM Core, we already had a subset created.Now we're going to create a new subset.Now we see we have a new subset model created, just for Emergency Management. And as we did before, we'll add the properties that we want for AlertType.We now have a subset of EM with juts the AlertType and some of its properties.This can be used in addition to the subsets we've defined from NIEM Core.

Extension models define new concepts not found in the reference models.These can be new classes, such as object types, roles, association, metadata, or adapters, new properties, augmentations, and new classes may or may not subset subclass concepts found in the reference models. Let's look at another example.

We'll create a new Extension Model for alerts.We'll add a subclass of AlertType in the extension, we'll add a new enumeration for the kind of alert, whether its silent or an alarm.And then we'll add an AlertKind property to AlertType.While we already have an extension model, we'll create another one by way of example.We'll create a package and look at it specification to add a stereotype. We'll now bring in AlertType from our subset and create a new class that extends it.We'd also like an enumeration to hold our AlarmType.What we'd like to do now is add a property to AlertType extension for our AlarmKind.We now have a proper extension of AlertType including our AlarmKind.

NIEM also defines the concept of augmentations.Augmentations allow new properties to be defined for a class across multiple Namespaces and then be arbitrarily combined.In this case, we see that TelephoneNumber and PetAdoption uses the Core definition of TelephoneNumber as well as an augmentation that includes TelephoneType. An augmentation is like any other class; however, it has the augmentation type stereotype.A class may then inherit any number of Augmentation types and the properties will be combined.Optionally, an Augmentation Type may define the class that it augments.In this case, TelephoneNumberAugmentation is restricted to only augment TelephoneNumbers.

Another way to restrict what an augmentation augments is at the property level. A property may have an augmentation application to a specific class.In this case, we see TelephoneNumberAugmentation is restricted to only augmenting TelephoneNumbers.However, other properties using TelephoneNumberAugmentation could augment other classes.

Let's go back to our example.We would like our AlertKind to be able to be arbitrarily included in any number of classes.So we're going to change our AlertType to an augmentation.We'll then define a new AlertType that inherits the initial AlertType, as well as this augmentation.We'll then explore the XML to see how this looks.In this example, we'll make AlarmKindCode more reusable.

We'll do that by changing this to an augmentation and appropriately renaming it.We'll now add a new AlertType that inherits both base AlertType and the augmentation.We also want to say the AlertAugmentationType augments AlertType.That means the AlarmKindCode properties only applies to AlertType.What we're going to do now is generate the MPD so it can look at the XML.The generation is complete, so let's look at one of the schemas.

Here we see the XML and we can identify the AlarmCode, the AugmentationType that uses the AlarmCode, and the AlertType referencing the property that includes the augmentation. In this way we can see that AlertType includes both the original definition of alert, as well as the augmented AlarmKindCode.

This concludes the NIEM tutorial on augmentation and extension.For more information on NIEM-UML, please see the webinars on the NIEM.gov website.Thank you for your interest in NIEM-UML.